**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный университет**

**Факультет прикладной математики-процессов управления**

**Кафедра “фундаментальная информатика и информационные технологии”**

**отчет**

**по лабораторной работе №5**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему:**

**«Разработка и реализация алгоритма роевого интеллекта для решения задач глобальной оптимизации»**

**Вариант – 6**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б15-пу |  | Колосков В.Ю. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г**

**Оглавление**

[**Цель работы** 3](#_Toc152038228)

[**Задачи** 3](#_Toc152038229)

[**Теоретическая часть** 3](#_Toc152038230)

[**Описание алгоритма** 5](#_Toc152038231)

[**Блок-схема программы** 6](#_Toc152038232)

[**Описание программы** 10](#_Toc152038233)

[**Рекомендации для пользователя** 12](#_Toc152038234)

[**Рекомендации для программиста** 15](#_Toc152038235)

[**Контрольный пример** 16](#_Toc152038236)

[**Анализ результатов работы алгоритма** 18](#_Toc152038237)

[**Вывод** 22](#_Toc152038238)

[**Список использованной литературы** 22](#_Toc152038239)

# **Цель работы**

Разработать программу, которая находит глобальный минимум функции при помощи алгоритма роя частиц.

# **Задачи**

1. Изучить особенности алгоритма роевого интеллекта
2. Разработать алгоритм поиск минимума функции при помощи алгоритма роя частиц
3. Написать программу, которая находит глобальный минимум заданной функции и найти минимум заданной функции
4. Проанализировать результаты работы алгоритма
5. Сравнить результат работы роевого интеллекта с результатами работы генетического алгоритма.

# **Теоретическая часть**

Алгоритм роя частиц – итерационный алгоритм оптимизации, находит оптимальное значения функции путем моделирования поведения роя.

В алгоритме роя частиц представляется группа частиц, каждая из которых представляет собой кандидатское решение. Каждая частица движется в пространстве поиска с определенной скоростью и запоминает свою лучшую позицию и лучшую позицию группы.

В каждой итерации алгоритм обновляет положение и скорость каждой частицы на основе ее текущего положения, лучшей позиции группы и лучшей позиции самой частицы. При этом частицы взаимодействуют друг с другом, обмениваясь информацией о своих лучших позициях, чтобы найти лучшее глобальное решение.

Для расчета скорости частицы используется формула:

, где:

– вектор скорости частицы

– Случайно значение в промежутке (0, 1)

– постоянная локального ускорения

– постоянная глобального ускорения

– точка локального оптимального значения

– точка глобального оптимального значения

– координаты точки частицы

Новая координата точке расчитывается по формуле:

, где:

– координаты точки частицы

– вектор скорости частицы

Алгоритм продолжается до достижения заданного критерия останова, такого как достижение определенного числа итераций или достижение заданного значения функции цели.

# **Описание алгоритма**

1. Инциализация роя со случайными параметрами частиц в заданных промежутках
2. Расчет наилучшего глобального значения роя и наилучшего локального значения каждой частицы
3. Нахождение вектора скорости каждого агента на основе глобального значения роя.
4. Нахождение новых координат агентов на основе исходных координат их скорости
5. Повторить шаги 2-4 до тех пор, пока не будет пройдено заданное число итераций
6. Вывести глобальный минимум функции и координаты точки, в которой он достигается

# 

# **Блок-схема программы**

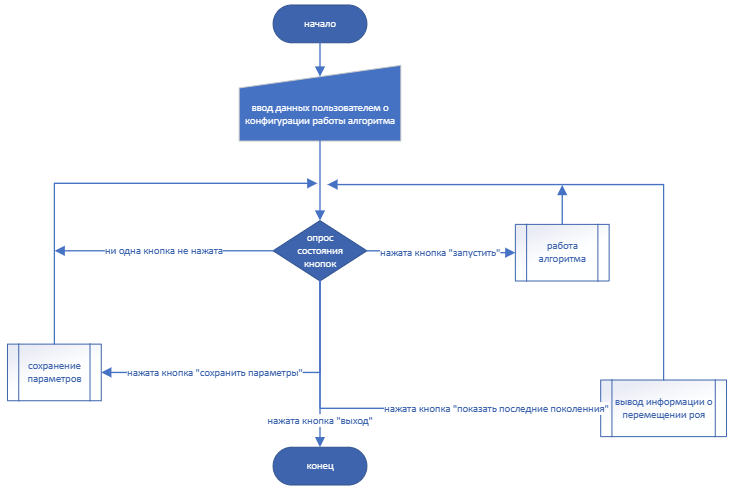


Рисунок 1. Основное тело программы

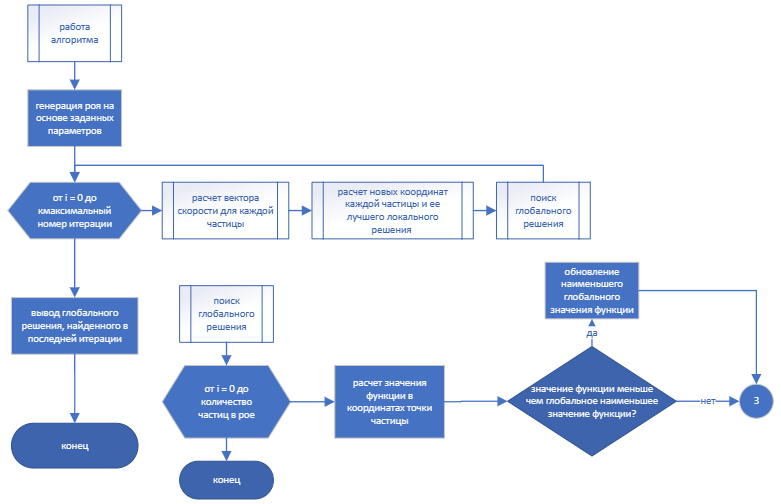


Рисунок 2. Алгоритм роя частиц. Функция поиска глобального решения

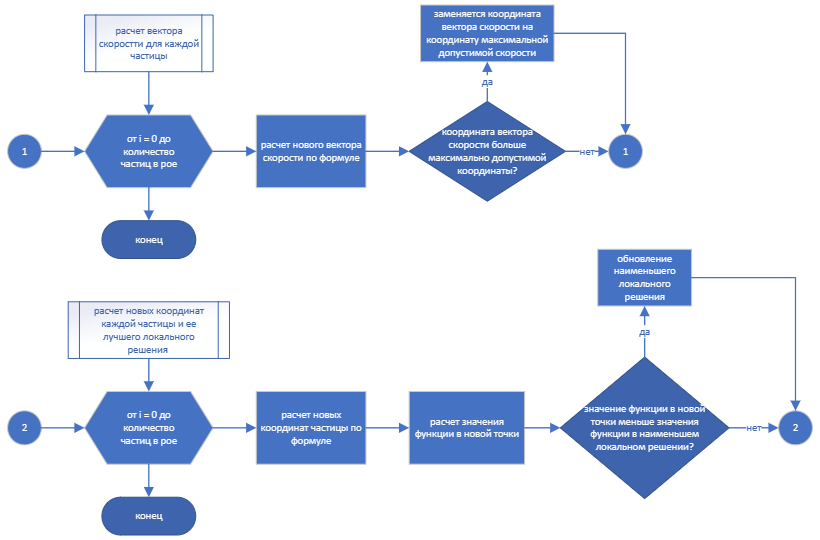


Рисунок 3. Функции: Расчет новых координат частиц, расчет вектора скорости частиц

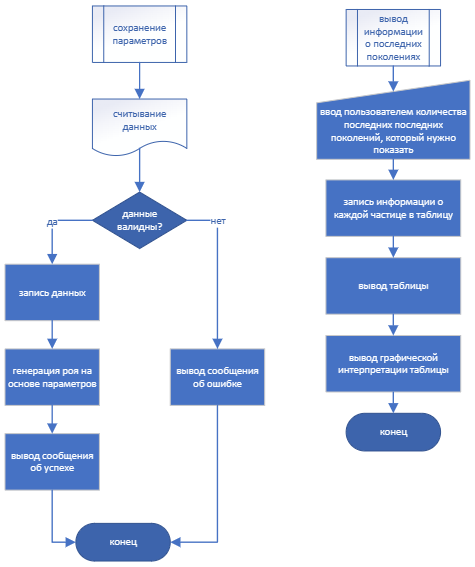


Рисунок 4. Функции: сохранение параметров, вывод информации о последних поколениях

# **Описание программы**

Программа реализована при помощи языка python 3.11.4, с использованием библиотек: os, tkinter, sys, numpy, random, matplotlib. В программе использовались 3 классa, 5 функций и 7 структур данных. В таблице 1 представлено описание методов класса Func.

Таблица 1. Описание методов класса Func

|  |  |  |
| --- | --- | --- |
| **Метод** | **Описание** | **Тип возвращаемого значения** |
| value | Рассчитывает значение функции в точке | Float |

В таблице 2 представлено описание методов класса Individual

Таблица 2. Описание методов класса Particle

|  |  |  |
| --- | --- | --- |
| **Метод** | **Описание** | **Тип возвращаемого значения** |
| Update\_v | Расчет вектора скорости частицы | None |
| Update\_x | Расчет координаты точки частицы | None |
| Update\_local\_solution | Расчет нового оптимального локального значения | None |
| clone | Копирование частицы | Particle |

В таблице 3 представлено описание методов класса GUI

Таблица 3. Описание методов класса GUI

|  |  |  |
| --- | --- | --- |
| **Метод** | **Описание** | **Тип возвращаемого значения** |
| start | Запуск графического интерфейса | None |
| Safe\_parametres | Сохранение параметров работы алгоритма, введенных пользователем | None |
| Start\_algoritm | Запуск алгоритма | None |
| show | Получение количества поколений, которые необходимо вывести, запись параметров в таблицу | None |

В таблице 4 представлено описание функций

Таблица 4. Описание функций

|  |  |  |
| --- | --- | --- |
| **Функция** | **Описание** | **Тип возвращаемого значения** |
| Search\_global\_solution | Поиск глобального оптимального решения | numpy.ndarray |
| Moving\_swarm | Расчет координат частиц и поиск их оптимального локального решения | Swarm |
| Chanding\_vector | Расчет вектора скорости для каждой частицы | Swarm |
| Create\_swarm | Инициализация начального роя | Swarm |
| main | Основная функция | None |

В таблице 5 представлено описание структур данных.

Таблица 5. Описание структур данных

|  |  |  |
| --- | --- | --- |
| **Имя структуры** | **Тип структуры** | **Хранимые данные** |
| v | Numpy.ndarray | Вектор скорости частицы |
| x | Numpy.ndarray | Коодината точки частицы |
| swarm | Swarm | Частицы в рое |
| Best\_coordinate | tuple | Координаты точки, в которой значение функции достигает минимума |
| const | Constants | Параметры работы алгоритма |

# **Рекомендации для пользователя**

Для запуска программы необходимо наличие устройства с установленной операционной системой Linux, macOS или Windows, а также среды разработки, поддерживающей запуск python версии 3.11.4.

Необходимо установить python библиотеки: moviepy, tkinter и matplotlib. Для этого необходимо открыть терминал и прописать следующие команды: pip install matplotlib, pip install tkinker, pip install moviepy.

После запуска приложения появится главное окно программы.

Вверху находится меню программы, с кнопками (рисунок 5).

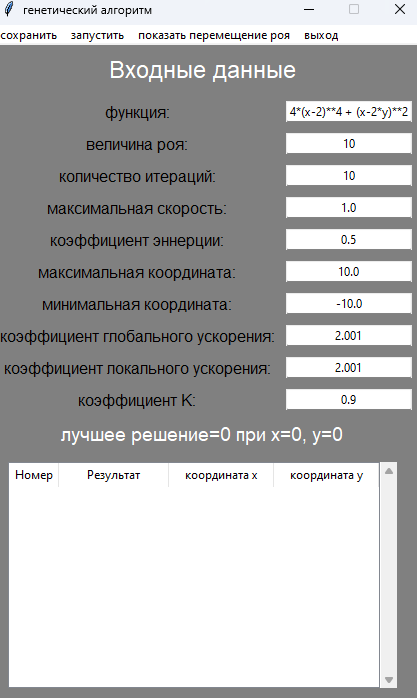


Рисунок 5. Меню программы

В таблице 7 представлено описание кнопок.

Таблица 7. Описание кнопок

|  |  |
| --- | --- |
| **Название кнопки** | **Описание операции, которая выполняется при нажатии на кнопку** |
| Сохранить | Сохраняет параметры, введенные пользователем |
| Запустить | Запуск алгоритма роя частиц |
| Показать перемещения роя | Отображает параметры роя последних выбранных перемещений |
| Выход | Выход из программы |

В волях ввода, расположенных под меню программы необходимо ввести функцию и параметры работы алгоритма (рисунок 6).

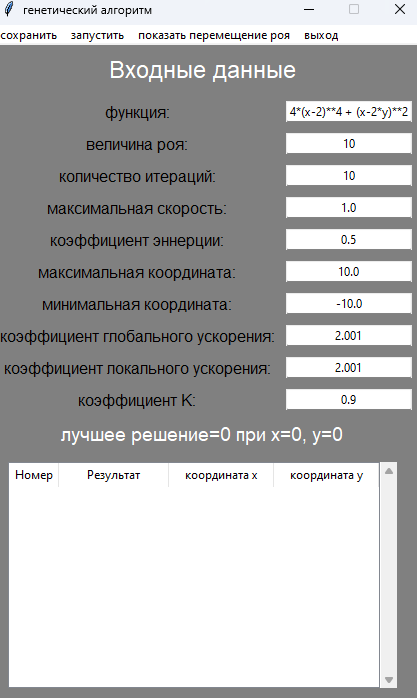


Рисунок 6. Поля ввода параметров

В строке, расположенной под полями ввода параметров после нажатия на кнопку «запустить» отобразится минимум функции и координаты точки, в которой он достигается при заданных параметрах (рисунок 7).

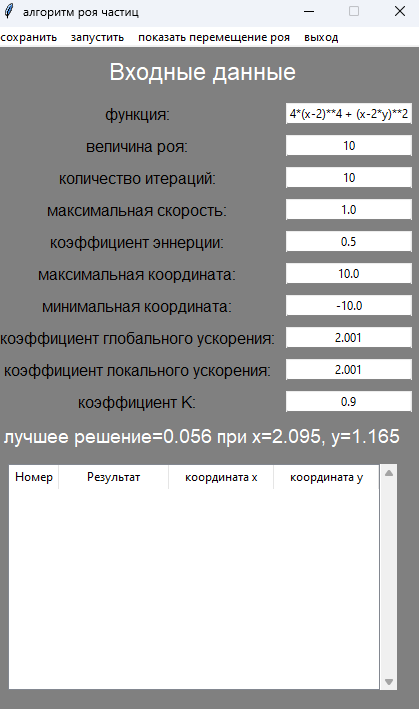


Рисунок 7. Результат работы алгоритма

После нажатия на кнопку «показать последнии поколения» В таблице отобразятся параметры особей последних выбранных поколений (рисунок 8).

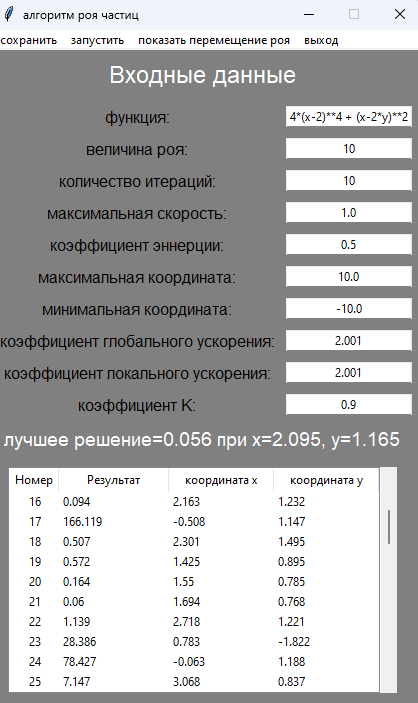


Рисунок 8. Особи последних выбранных поколений

И появится график, на котором продемонстрировано распределения особей популяций (рисунок 9).

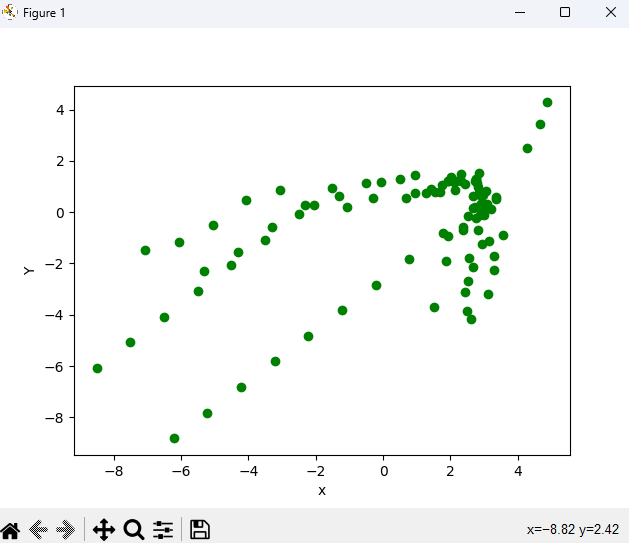
****

Рисунок 9. Распределение особей популяции

# **Рекомендации для программиста**

Для внесения изменений в исходный код программы необходимо наличие устройство с установленное операционной системой Linux, macOS или Windows, а также среды разработки, поддерживающей запуск python версии 3.11.4

Исходный код программы доступен по ссылке ниже:

[ARnoD0rian/ROI: ROI algoritm (github.com)](https://github.com/ARnoD0rian/ROI)

# **Контрольный пример**

В этом разделе представлен пример работы программы.

На рисунке 10 представлены параметры, установленные до начала работы программы.

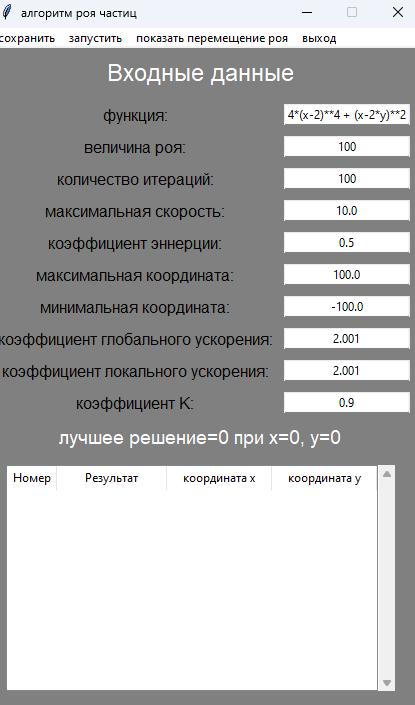


Рисунок10.Входные данные.

На рисунке 11 представлен результат работы программы.

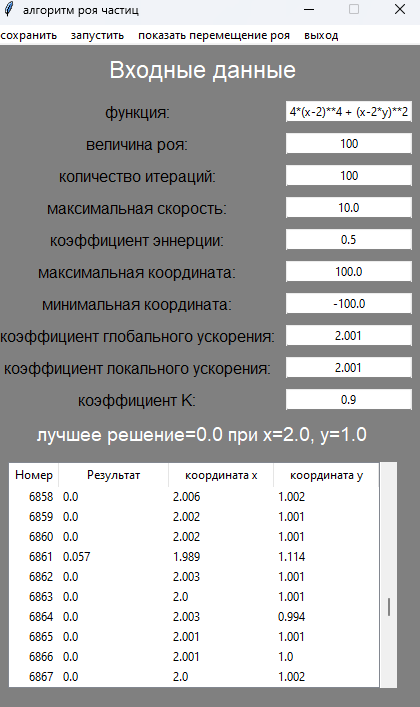


Рисунок 11. Результат работы программы

На рисунке 12 представлено распределение всех особей 100 поколений.

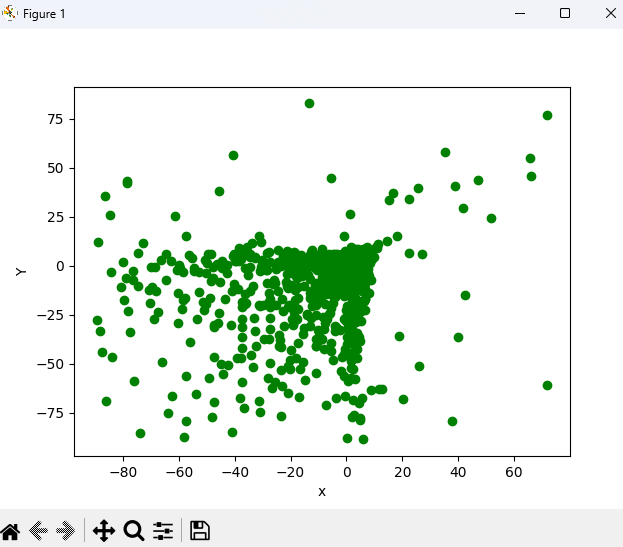


Рисунок 12. Распределение роя после 100 итераций

При работе алгоритма сохранились графические изображения распределения роя на каждой итерации. И создалось видео, которое демонстрирует работу алгоритма роя частиц (рисунок 13)

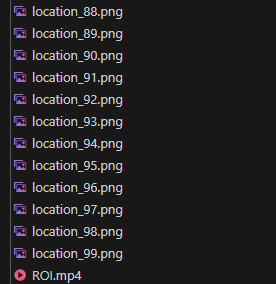


Рисунок 13. Сохранение изображений и видео, демонстрирующих работу алгоритма

# **Анализ результатов работы алгоритма**

Рассмотрим каноническую модификацию алгоритма роя частиц.

При такой модификации формула расчета вектора скорости имеет вид:

, где:

X – коэффициент сжатия и рассчитывается по формуле:

W – коэффициент инерции

При использовании данной модификации скорость работы алгоритма не изменится. Рассмотрим, как при данной модификации изменилась точность решения (рисунок 14).

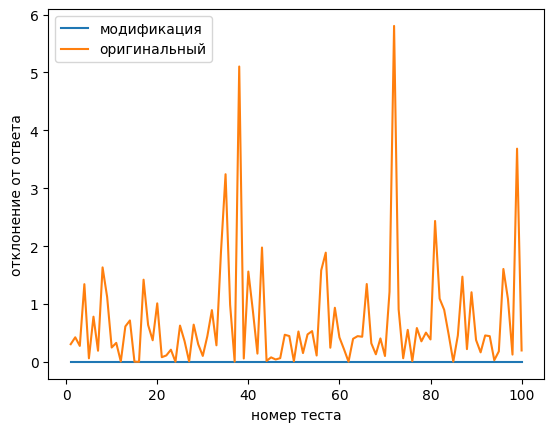


Рисунок 14. Сравнение модификации и оригинального алгоритма

Коэффициент инерции может быть как постоянный значением, так и зависеть от какой-то переменной. Пусть он зависит от номера итерации. Рассмотрим, как изменится точность решения при использовании различных функций (рисунок 15).

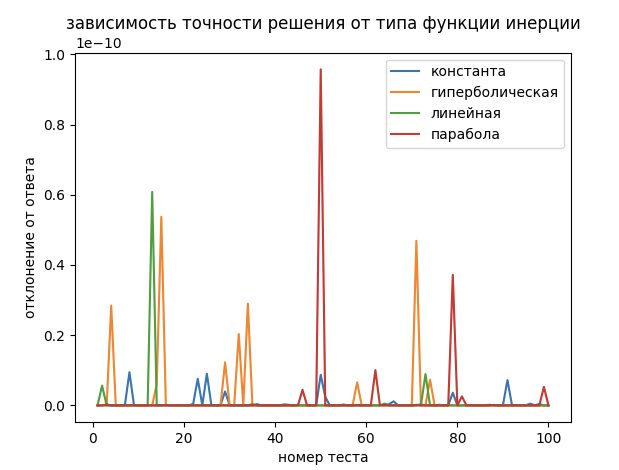


Рисунок 15. Зависимость точности решения от типа функции инерции

Из графика видно, что наименьшая точность достигается при использовании в качестве функции инерции параболы и линейной функции. Наименьшие возмущения достигаются при использовании в качестве функции инерции постоянного значения.

Из этого можно сделать вывод, что наибольшая точность в решении достигается при использовании в качестве функции инерции постоянного значения.

Рассмотрим, как изменяется точность алгоритма, при изменении количества итераций и величины роя. В таблицaх 8, 9, 10 представлены результаты тестов

Таблица 8. Изменение точности, при изменении количества итераций

|  |  |  |
| --- | --- | --- |
| **Количество итераций** | **Отклонение от ответа** | **Координаты точки** |
| 10 | 0.1614183722 | x=1.599, y=0.92 |
| 20 | 1.7072e-06 | x=1.976, y=0.988 |
| 30 | 7.99e-07 | x=2.021, y=1.01 |
| 40 | 3.129e-07 | x=2.017, y=1.008 |
| 50 | 1.771e-07 | x=2.014, y=1.007 |
| 60 | 1.287e-07 | x=2.013, y=1.007 |
| 70 | 1.078e-07 | x=2.013, y=1.006 |
| 80 | 9.27e-08 | x=2.012, y=1.006 |
| 90 | 7.92e-08 | x=2.012, y=1.006 |
| 100 | 7.09e-08 | x=2.011, y=1.006 |

Таблица 9. Изменение точности, при изменении размера роя

|  |  |  |
| --- | --- | --- |
| **Размер роя** | **Отклонение от ответа** | **Координаты точки** |
| 10 | 3823.4442207506 | x=4.941, y=-27.212 |
| 20 | 0.0396289789 | x=2.15, y=1.172 |
| 30 | 0.0461411947 | x=1.694, y=0.9 |
| 40 | 0.0161384842 | x=2.252, y=1.126 |
| 50 | 0.1971382186 | x=1.682, y=0.643 |
| 60 | 0.0022173133 | x=2.008, y=1.027 |
| 70 | 0.0032745027 | x=2.169, y=1.083 |
| 80 | 0.0012444487 | x=1.931, y=0.983 |
| 90 | 0.0016240393 | x=1.944, y=0.952 |
| 100 | 0.0044624867 | x=1.83, y=0.898 |

Таблица 10. Изменение точности, при изменении размера роя и количества итераций

|  |  |  |
| --- | --- | --- |
| **Размер роя и количество итераций** | **Отклонение от ответа** | **Координаты точки** |
| 10 | 41.79678938418486 | x=1.5853813416,  y=-2.4352567836 |
| 20 | 0.00012934544272714949 | x=1.936718419, y=0.9723965195 |
| 30 | 2.9285185839495934e-05 | x=1.9643782733, y=0.9845789422 |
| 40 | 1.0090096681385659e-11 | x=1.9989080255, y=0.9994550619 |
| 50 | 6.6269249736499235e-12 | x=1.9988654858, y=0.999432736 |
| 60 | 1.5079348015051931e-15 | x=2.000139328, y=1.0000696636 |
| 70 | 5.201061010687681e-23 | x=1.9999981074, y=0.9999990537 |
| 80 | 8.918675186414007e-17 | x=2.00006868, y=1.0000343402 |
| 90 | 1.5991719349009335e-22 | x=2.0000025144, y=1.0000012572 |
| 100 | 1.6302226953590146e-30 | x=1.9999999822, y=0.9999999911 |

Из этого можно сделать вывод, что при увеличении количества итераций точность алгоритма хорошо повышается. При увеличении размера роя, точность алгоритма повышается не очень сильно, это связано с тем, что частицам не хватает шагов, чтобы дойти до решения. При увеличении количества итераций и размера роя точность значительно повышается, при этом точность решения выше, чем при изменении только количества итераций. Это происходит из-за того, что при увеличении только итераций, алгоритму не хватает достаточного количества частиц.

# **Сравнение генетического алгоритма и алгоритма роя частиц**

Сравним оригинальный алгоритм роя частиц и генетический алгоритм с вещественным кодированием. На рисунке 16 показано время работы генетического алгоритма и алгоритма роя частиц.



Рисунок 16.Время работы генетического алгоритма и алгоритма роя частиц

Из графика видно, что алгоритм роя частиц работает медленнее, чем генетический алгоритм. Сравним два алгоритма на точность получаемого ответа (рисунок 17).

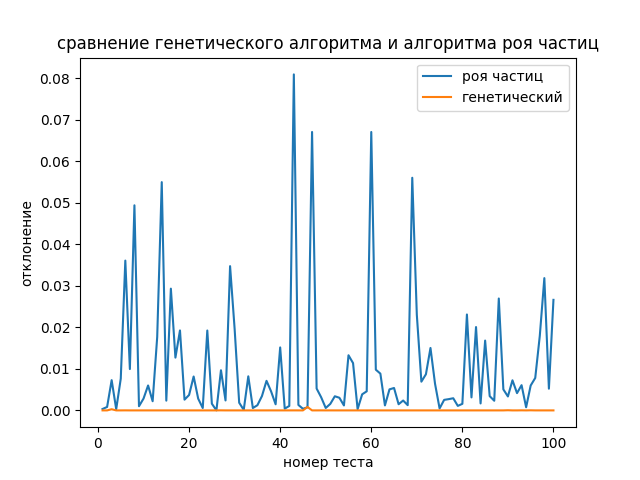


Рисунок 17. Отклонение от ответа генетического алгоритма и алгоритма роя частиц

Из графика видно, что генетический алгоритм дает более точный ответ. Из этого можно сделать вывод, что генетический алгоритм более эффективный, чем оригинальный алгоритм роя частиц.

Сравним модификацию алгоритма роя и генетического алгоритма. На рисунке 18 показано время работы алгоритмов.

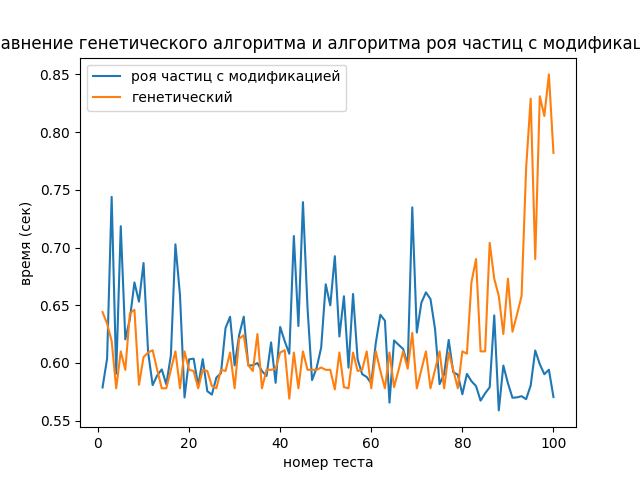


Рисунок 18. Сравнение во времени работы генетического алгоритма и алгоритма роя частиц

График демонстрирует что их скорость работы примерно одинакова. Сравним эти алгоритмы на точность вычислений (рисунок 19).

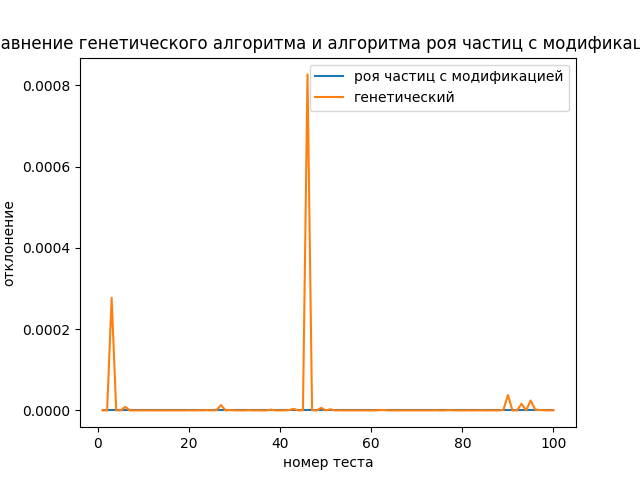
****

Рисунок 19.Сравнение алгоритма роя частиц с модификацией и генетического алгоритма в точности вычислений.

Из графика видно, что генетический алгоритм менее точный, чем алгоритм роя частиц. Это связано с тем, что в генетическом алгоритме есть мутации, которые могут отклонить от правильного решения.

# **Листинг программы**

На рисунках 20 – 22 представлен листинг кода алгоритма с подробными комментариями.

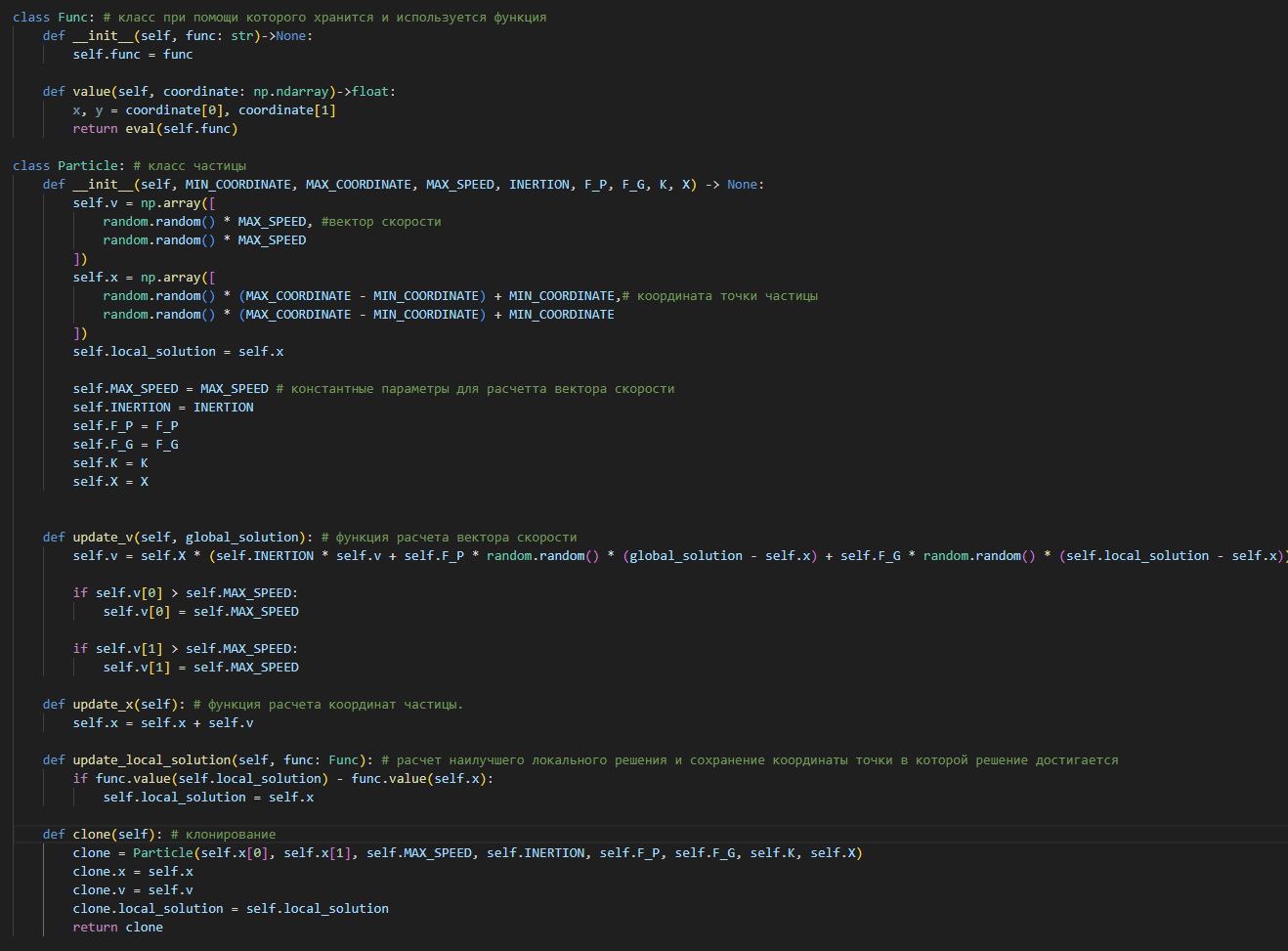


Рисунок 20. Класс функции и класс частицы

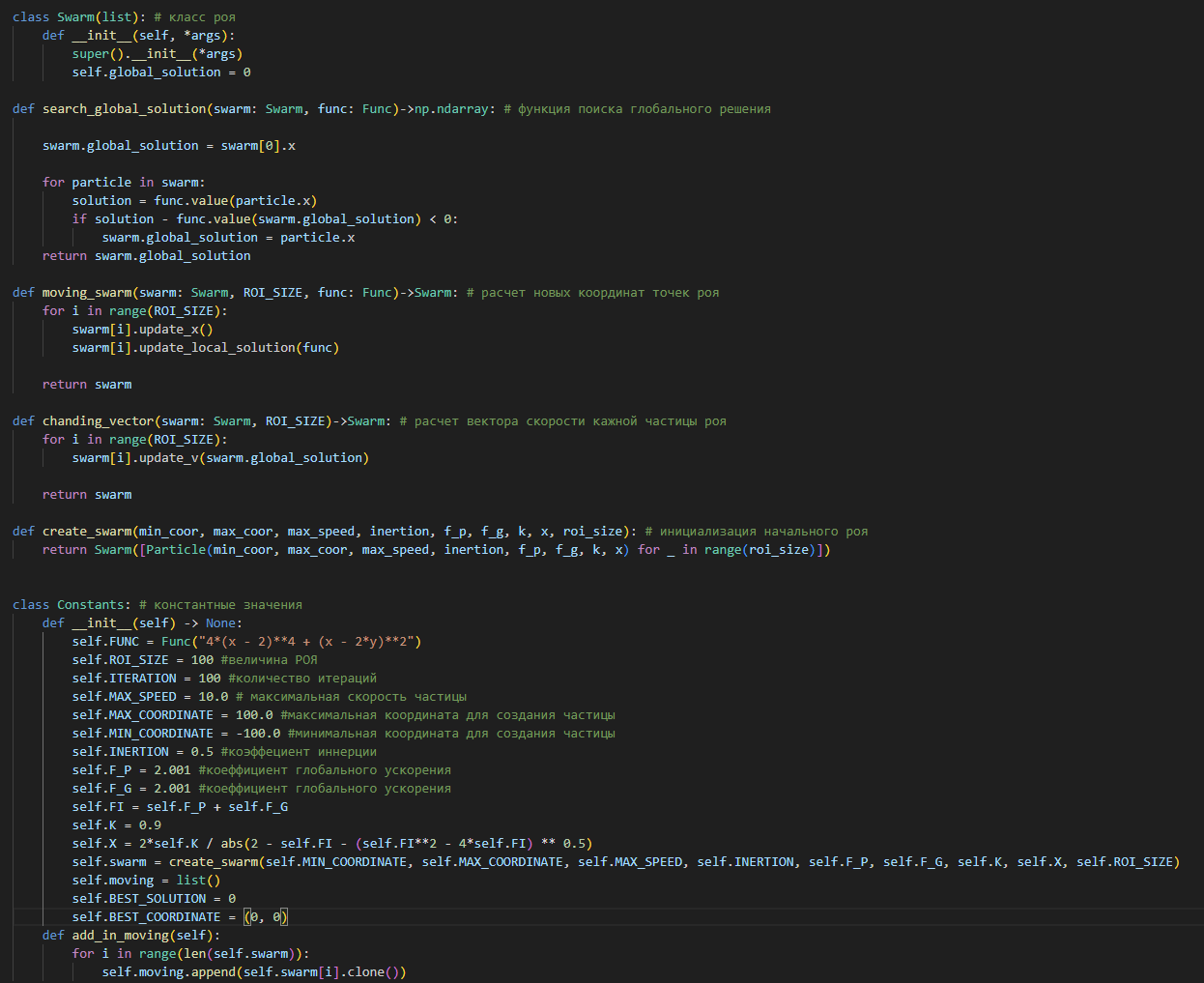


Рисунок 21. Класс Роя, класс константный значений. Функции: инициализация роя, расчет новых векторов скорости частиц, расчет новых координат частиц

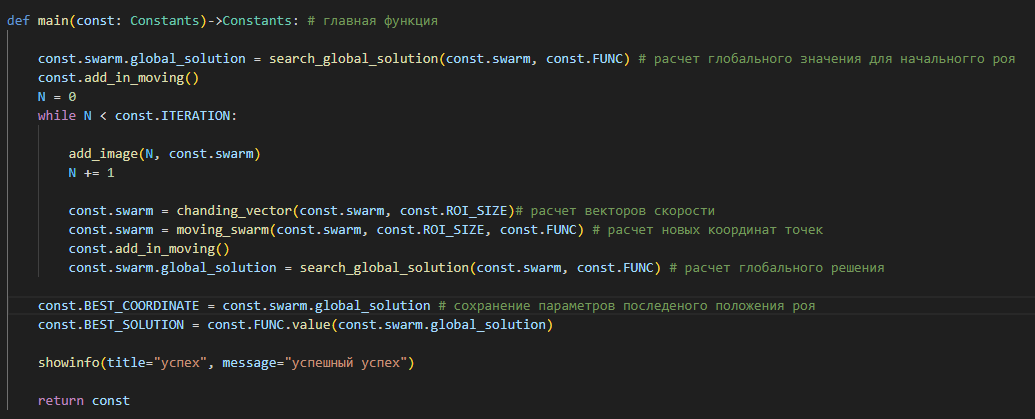


Рисунок 22. Основная функция

# **Вывод**

В ходе выполнения данной работы были изучены особенности роевого интеллекта. Разработан алгоритм поиска минимума функции при помощи алгоритма роя частиц. Написана программа, которая находит глобальный минимум функции при помощи алгоритма роя частиц. Найден глобальный минимум функции:

равный 0 и достигается в точке (2, 1)

Предложена модификация канонического алгоритма для алгоритма роя частиц и показано, что модификация быстрее и точнее определяет глобальный минимум функции.

Было проведено сравнение результатов работы алгоритмов Роя частиц и генетического алгоритма. И показано, что генетический алгоритм быстрее и точнее чем оригинальный роевой алгоритм, но менее точный, чем алгоритм роя частиц с канонической модификацией.

# **Список использованной литературы**

[1] Статья о роевом интеллекте: [Роевой интеллект — Википедия (wikipedia.org)](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B5%D0%B2%D0%BE%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82)

[2] статья о канонической модификации роя: [Алгоритм роя частиц / Хабр (habr.com)](https://habr.com/ru/articles/105639/)

[2] Документация tkinter: [Руководство по программированию на Tkinter и Python (metanit.com)](https://metanit.com/python/tkinter/)